What is claimed is:

1. A method for use with those of a plurality of commands sent by at least one client on a network connected to a network communications coupler designated by said at least one client as delayed response commands comprising the steps of:

storing in said coupler a first instance of a template, said first instance used to store in said coupler at least one of said delayed response commands during execution of said at least one delayed response command by said coupler; and

storing in said coupler a second instance of said template, said second instance used to store in said coupler a reply to said at least one delayed response command executed by said coupler until said reply is retrieved by said client.

2. The method of Claim 1 wherein said template has a declaration having one template parameter class in the parameter list of said template and one or more statements defining said template.

3. The method of Claim 2 wherein said one or more statements defining said template include a pointer of said template parameter class.

4. The method of Claim 3 wherein said one or more statements defining said template further include enumerated special values for designating boundary and special conditions and one or more methods.

5. The method of Claim 4 wherein said one or more methods statements comprise constructor, destructor and access methods.

6. The method of Claim 2 wherein said template is:

```
template <class Entry>
KeyedStorage<Entry>::KeyedStorage()
{
   storage = new Entry [1+MAXKEYVAL];

   {
      storage[i] = NULL;
   }
```

```
};

template <class Entry>
KeyedStorage<Entry>::~KeyedStorage()
{
   delete[] storage;
};

template <class Entry> int KeyedStorage<Entry>::putEntry(
int key, Entry newEntry )
{
   int success;

   if( key >=0 && key <= MAXKEYVAL )
   {
      storage[key] = newEntry;
      success = TRUE;
   }
   else
   {
      success = FALSE;
   }

   return( success );
};

template <class Entry> Entry
KeyedStorage<Entry>::getEntry( int key )
{
   Entry foundEntry;

   if( key >= 0 && key <= MAXKEYVAL )
   {
      foundEntry = storage[key];
      storage[key] = NULL;
   }
   else
   {
      foundEntry = NULL;
   }

   return( foundEntry );
};

template <class Entry> Entry
KeyedStorage<Entry>::checkEntry( int key )
{
   Entry foundEntry;

   if( key >= 0 && key <= MAXKEYVAL )
   {
      foundEntry = storage[key];
   }
   else
```

```
    {
        foundEntry = NULL;
    }

    return( foundEntry );
};

int KSTest(void )
{
    KeyedStorage<CNIReply *> ReplyStorage;
    KeyedStorage<CNICommand *> ActiveCommands;
    KeyedStorage<ClientInterface *> ClientInterfaces;

    CNIReply * Reply1 = (CNIReply *)1;
    CNIReply * Reply2 = (CNIReply *)2;

    CNICommand *Command1 = (CNICommand *)101;
    CNICommand *Command2 = (CNICommand *)102;

    ClientInterface *Client1 = (ClientInterface *)1001;
    ClientInterface *Client2 = (ClientInterface *)1002;

    ReplyStorage.putEntry( 1, Reply1 );
    ActiveCommands.putEntry( 1, Command1 );
    ClientInterfaces.putEntry( 1, Client1 );

    ReplyStorage.putEntry( 2, Reply2 );
    ActiveCommands.putEntry( 2, Command2 );
    ClientInterfaces.putEntry( 2, Client2 );

    printf("\n\ralles put\n\r");

    printf("\n\rchecking first reply = %p",
ReplyStorage.checkEntry( 1 ) );
    printf("\n\rchecking second reply = %p",
ReplyStorage.checkEntry( 2 ) );

    printf("\n\rchecking first command = %p",
ActiveCommands.checkEntry( 1 ) );
    printf("\n\rchecking second command = %p",
ActiveCommands.checkEntry( 2 ) );

    printf("\n\rchecking first client = %p",
ClientInterfaces.checkEntry( 1 ) );
    printf("\n\rchecking second client = %p",
ClientInterfaces.checkEntry( 2 ) );

    printf("\n\rgetting first reply = %p",
ReplyStorage.getEntry( 1 ) );
    printf("\n\rgetting first reply again = %p",
ReplyStorage.getEntry( 1 ) );

    printf("\n\rgetting second command = %p",
ActiveCommands.getEntry( 2 ) );
```

```
   printf("\n\rgetting second command again = %p",
ActiveCommands.getEntry( 2 ) );

   printf("\n\rgetting first client = %p",
ClientInterfaces.getEntry( 1 ) );
   printf("\n\rgetting first client again = %p",
ClientInterfaces.getEntry( 1 ) );

   printf("\n\r");

   return 0;
}
```

7. The method of Claim 1 further comprising the step of defining for said template a declaration having one template parameter class in the parameter list of said template and one or more statements defining said template.

8. The method of Claim 7 further comprising the step of defining for said one or statements a pointer of said one template parameter class, enumerated special values for designating boundary and special conditions, and one or more methods.

9. The method of 8 further comprising the step of defining for said one or statements that are method statements methods selected from constructor, destructor and access methods.

10. A method for use with commands sent to a network communications coupler by at least one client on a network connected to said coupler comprising the steps of:

designating by said at least one client which of said commands are delayed response commands,

storing in said coupler a first instance of a template, said first instance used to store in said coupler at least one of said delayed response commands during execution of said at least one delayed response command by said coupler; and

storing in said coupler a second instance of said template, said second instance used to store in said coupler a reply to said at least one delayed response

command executed by said coupler until said reply is retrieved by said client.

11. The method of Claim 10 wherein said template has a declaration having one template parameter class in the parameter list of said template and one or more statements defining said template.

12. The method of Claim 11 wherein said one or more statements defining said template include a pointer of said template parameter class.

13. The method of Claim 12 wherein said one or more statements defining said template further include enumerated special values for designating boundary and special conditions and one or more methods.

14. The method of Claim 13 wherein said one or more methods statements comprise constructor, destructor and access methods.

15. The method of Claim 11 wherein said template is:

```
template <class Entry>
KeyedStorage<Entry>::KeyedStorage()
{
   storage = new Entry [1+MAXKEYVAL];

   {
      storage[i] = NULL;
   }
};

template <class Entry>
KeyedStorage<Entry>::~KeyedStorage()
{
   delete[] storage;
};

template <class Entry> int KeyedStorage<Entry>::putEntry(
int key, Entry newEntry )
{
   int success;

   if( key >=0 && key <= MAXKEYVAL )
   {
      storage[key] = newEntry;
      success = TRUE;
   }
   else
```

```
    {
        success = FALSE;
    }

    return( success );
};

template <class Entry> Entry
KeyedStorage<Entry>::getEntry( int key )
{
    Entry foundEntry;

    if( key >= 0 && key <= MAXKEYVAL )
    {
        foundEntry = storage[key];
        storage[key] = NULL;
    }
    else
    {
        foundEntry = NULL;
    }

    return( foundEntry );
};

template <class Entry> Entry
KeyedStorage<Entry>::checkEntry( int key )
{
    Entry foundEntry;

    if( key >= 0 && key <= MAXKEYVAL )
    {
        foundEntry = storage[key];
    }
    else
    {
        foundEntry = NULL;
    }

    return( foundEntry );
};

int KSTest(void )
{
    KeyedStorage<CNIReply *> ReplyStorage;
    KeyedStorage<CNICommand *> ActiveCommands;
    KeyedStorage<ClientInterface *> ClientInterfaces;

    CNIReply * Reply1 = (CNIReply *)1;
    CNIReply * Reply2 = (CNIReply *)2;

    CNICommand *Command1 = (CNICommand *)101;
    CNICommand *Command2 = (CNICommand *)102;
```

```
   ClientInterface *Client1 = (ClientInterface *)1001;
   ClientInterface *Client2 = (ClientInterface *)1002;

   ReplyStorage.putEntry( 1, Reply1 );
   ActiveCommands.putEntry( 1, Command1 );
   ClientInterfaces.putEntry( 1, Client1 );

   ReplyStorage.putEntry( 2, Reply2 );
   ActiveCommands.putEntry( 2, Command2 );
   ClientInterfaces.putEntry( 2, Client2 );

   printf("\n\ralles put\n\r");

   printf("\n\rchecking first reply = %p",
ReplyStorage.checkEntry( 1 ) );
   printf("\n\rchecking second reply = %p",
ReplyStorage.checkEntry( 2 ) );

   printf("\n\rchecking first command = %p",
ActiveCommands.checkEntry( 1 ) );
   printf("\n\rchecking second command = %p",
ActiveCommands.checkEntry( 2 ) );

   printf("\n\rchecking first client = %p",
ClientInterfaces.checkEntry( 1 ) );
   printf("\n\rchecking second client = %p",
ClientInterfaces.checkEntry( 2 ) );

   printf("\n\rgetting first reply = %p",
ReplyStorage.getEntry( 1 ) );
   printf("\n\rgetting first reply again = %p",
ReplyStorage.getEntry( 1 ) );

   printf("\n\rgetting second command = %p",
ActiveCommands.getEntry( 2 ) );
   printf("\n\rgetting second command again = %p",
ActiveCommands.getEntry( 2 ) );

   printf("\n\rgetting first client = %p",
ClientInterfaces.getEntry( 1 ) );
   printf("\n\rgetting first client again = %p",
ClientInterfaces.getEntry( 1 ) );

   printf("\n\r");

   return 0;
}
```

16. The method of Claim 10 further comprising the step of defining for said template a declaration having one template parameter class in the parameter list of said template and one or more statements defining said template.

17. The method of Claim 16 further comprising the step of defining for said one or statements a pointer of said one template parameter class, enumerated special values for designating boundary and special conditions, and one or more methods.

18. The method of 17 further comprising the step of defining for said one or statements that are method statements methods selected from constructor, destructor and access methods.